Package: HMMpa (via r-universe)

March 5, 2025

Type Package

Title Analysing Accelerometer Data Using Hidden Markov Models

Version 1.0.2

Date 2025-01-27

Description Analysing time-series accelerometer data to quantify length and intensity of physical activity using hidden Markov models. It also contains the traditional cut-off point method. Witowski V, Foraita R, Pitsiladis Y, Pigeot I, Wirsik N (2014). <doi:10.1371/journal.pone.0114089>.

Depends R (>= 2.10.0)

Imports graphics, stats

URL https://github.com/bips-hb/HMMpa

BugReports https://github.com/bips-hb/HMMpa/issues

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Repository https://bips-hb.r-universe.dev

RemoteUrl https://github.com/bips-hb/HMMpa

RemoteRef HEAD

RemoteSha 7f2ddcc4d8a4cefc555af091adfee09df4ae1a28

Contents

`HMMpa-package` *Analysing Accelerometer Data Using Hidden Markov Models*

Description

This package provides functions for analyzing accelerometer output data (known as a time-series of (impulse)-counts) to quantify length and intensity of physical activity.

Details

Usually, so called *activity ranges* are used to classify an activity as "sedentary", "moderate" and so on. *Activity ranges* are separated by certain thresholds (*cut-off points*). The choice of these cut-off points depends on different components like the subjects' age or the type of accelerometer device.

Cut-off point values and defined activity ranges are important input values of the following analyzing tools provided by this package:

1. **Cut-off point method**: Assigns an activity range to a count based on its total magnitude independently of other counts.

2. **HMM-based method**: Assigns an activity range to a count using a stochastic hidden Markov model (HMM), which identifies the physical activity states underlying the given time-series.

The HMM procedure for analyzing accelerometer data can be summarized as follows:

1. Train an HMM to estimate the number of hidden physical activity states (`m`) and model parameters (`delta`, `gamma`, `distribution_theta`).

2. Decode the trained HMM to classify accelerometer counts into `m` states.

3. Assign activity ranges based on the total magnitudes of the corresponding states.

Author(s)

Maintainer: Foraita Ronja <foraita@leibniz-bips.de> (ORCID)

Authors:

- Vitali Witowski

References

Witowski, V., Foraita, R., Pitsiladis, Y., Pigeot, I., Wirsik, N. (2014) Using hidden Markov models to improve quantifying physical activity in accelerometer data - A simulation study. PLOS ONE. *9*(12), e114089. doi:10.1371/journal.pone.0114089

See Also

Useful links:

- https://github.com/bips-hb/HMMpa
- Report bugs at https://github.com/bips-hb/HMMpa/issues

Examples

```
x <- c(1,16,19,34,22,6,3,5,6,3,4,1,4,3,5,7,9,8,11,11,
  14,16,13,11,11,10,12,19,23,25,24,23,20,21,22,22,18,7,
  5,3,4,3,2,3,4,5,4,2,1,3,4,5,4,5,3,5,6,4,3,6,4,8,9,12,
  9,14,17,15,25,23,25,35,29,36,34,36,29,41,42,39,40,43,
  37,36,20,20,21,22,23,26,27,28,25,28,24,21,25,21,20,21,
  11,18,19,20,21,13,19,18,20,7,18,8,15,17,16,13,10,4,9,
  7,8,10,9,11,9,11,10,12,12,5,13,4,6,6,13,8,9,10,13,13,
  11,10,5,3,3,4,9,6,8,3,5,3,2,2,1,3,5,11,2,3,5,6,9,8,5,
  2,5,3,4,6,4,8,15,12,16,20,18,23,18,19,24,23,24,21,26,
  36,38,37,39,45,42,41,37,38,38,35,37,35,31,32,30,20,39,
  40,33,32,35,34,36,34,32,33,27,28,25,22,17,18,16,10,9,
  5,12,7,8,8,9,19,21,24,20,23,19,17,18,17,22,11,12,3,9,
  10,4,5,13,3,5,6,3,5,4,2,5,1,2,4,4,3,2,1)


# Traditional Cut-off-point method ------------------------
traditional_cut_off_point_method <-
   cut_off_point_method(
               x = x,
               cut_points = c(5,15,23),
               names_activity_ranges = c("SED","LIG","MOD","VIG"),
               bout_lengths = c(1,1,2,4,5,10,11,20,21,60,61,260),
               plotting = 1)


# HMM-based Cut-off-point method --------------------------
# Use a (m = 4 state) hidden Markov model based on the
# generalized poisson distribution to assign an
# activity range to the counts.

# In this example three activity ranges
# (named as "light", "moderate" and "vigorous" physical activity)
# are separated by the two cut-points 15 and 23.


HMM_based_cut_off_point_method <-
   HMM_based_method(
               x = x,
```

```
                cut_points = c(15,23),
                min_m = 4,
                max_m = 4,
                names_activity_ranges = c("LIG","MOD","VIG"),
                distribution_class = "genpois",
                training_method = "numerical",
                DNM_limit_accuracy = 0.05,
                DNM_max_iter = 10,
                bout_lengths = c(1,1,2,4,5,10,11,20,21,60,61,260),
                plotting = 1)


# The HMM-based approach can be split into three steps ---------
# 1) Training of a HMM for given time-series of accelerometer counts
#    Here: A poisson distribution is trained based on a HMM for
#          m = 2,..., 6 states.
#          Select the HMM with the most plausibel m.

      m_trained_HMM <- HMM_training(x = x,
                                    min_m = 2,
                                    max_m = 6,
                distribution_class = "pois")$trained_HMM_with_selected_m


# 2) Decoding the trained HMM to extract hidden physical
#    activity (PA) levels

hidden_PA_levels <- HMM_decoding(x = x,
                                   m = m_trained_HMM$m,
                                delta = m_trained_HMM$delta,
                                gamma = m_trained_HMM$gamma,
                 distribution_class = m_trained_HMM$distribution_class,
                 distribution_theta = m_trained_HMM$distribution_theta)
 hidden_PA_levels <- hidden_PA_levels$decoding_distr_means


# 3) Assigning user-specified activity ranges to the accelerometer
#    counts via the total magnitudes of their corresponding
#    hidden PA-level
#    Here: 4 activity levels ("sedentary", "light", "moderate" and
#          "vigorous" physical activity) are separated by
#           3 cut-point (5, 15, 23)

 HMM_based_cut_off_point_method <-
    cut_off_point_method(x = x,
          hidden_PA_levels = hidden_PA_levels,
                cut_points = c(5,15,23),
     names_activity_ranges = c("SED","LIG","MOD","VIG"),
               bout_lengths = c(1,1,2,4,5,10,11,20,21,60,61,260),
                   plotting = 1)
```

```
# Simulate data of a large time-series of highly scattered counts ----
x <- HMM_simulation(size = 1500,
                    m = 10,
                    gamma = 0.93 * diag(10) + rep(0.07 / 10, times = 10),
                    distribution_class = "norm",
                    distribution_theta = list(mean = c(10, 100, 200, 300, 450,
                       600, 700, 900, 1100, 1300, 1500),
                    sd = c(rep(100,times=10))),
                    obs_round=TRUE,
                    obs_non_neg=TRUE,
                    plotting=5)$observations

# Compare results of the tradional cut-point method
# and the (6-state-normal-)HMM based method

traditional_cut_off_point_method <-
   cut_off_point_method(x = x,
                        cut_points = c(200,500,1000),
                        names_activity_ranges = c("SED","LIG","MOD","VIG"),
                        bout_lengths = c(1,1,2,4,5,10,11,20,21,60,61,260),
                        plotting = 1)

HMM_based_cut_off_point_method <-
   HMM_based_method(x = x,
                    max_scaled_x = 200,
                    cut_points = c(200,500,1000),
                    min_m = 6,
                    max_m = 6,
                    BW_limit_accuracy = 0.5,
                    BW_max_iter = 10,
                    names_activity_ranges = c("SED","LIG","MOD","VIG"),
                     distribution_class = "norm",
                     bout_lengths = c(1,1,2,4,5,10,11,20,21,60,61,260),
                     plotting = 1)
```

AIC_HMM *AIC Value for a Discrete Time Hidden Markov Model*

Description

Computes the Aikaike's information criterion and the Bayesian information criterion for a discrete time hidden Markov model, given a time-series of observations.

Usage

```
AIC_HMM(logL, m, k)
```

Arguments

logL	logarithmized likelihood of the model
m	integer number of states in the Markov chain of the model
k	single integer value representing the number of parameters of the underlying distribution of the observation process (e.g. k=2 for the normal distribution (mean and standard deviation))

Details

For a discrete-time hidden Markov model, AIC and BIC are as follows (MacDonald & Zucchini (2009, Paragraph 6.1 and A.2.3)):

$$\text{AIC} = -2logL + 2p$$

$$\text{BIC} = -2logL + p \log T$$

where T indicates the length/size of the observation time-series and p denotes the number of independent parameters of the model. In case of a HMM as provided by this package, where k and m are defined as in the arguments above, p can be calculated as follows:

$$p = m^2 + km - 1.$$

Value

The AIC or BIC value of the fitted hidden Markov model.

Author(s)

Based on MacDonald & Zucchini (2009, Paragraph 6.1 and A.2.3). Implementation by Vitali Witowski (2013).

References

MacDonald, I. L., Zucchini, W. (2009) *Hidden Markov Models for Time Series: An Introduction Using R*, Boca Raton: Chapman & Hall.

See Also

BIC_HMM, HMM_training

Examples

```
x <- c(1,16,19,34,22,6,3,5,6,3,4,1,4,3,5,7,9,8,11,11,
  14,16,13,11,11,10,12,19,23,25,24,23,20,21,22,22,18,7,
  5,3,4,3,2,3,4,5,4,2,1,3,4,5,4,5,3,5,6,4,3,6,4,8,9,12,
  9,14,17,15,25,23,25,35,29,36,34,36,29,41,42,39,40,43,
  37,36,20,20,21,22,23,26,27,28,25,28,24,21,25,21,20,21,
  11,18,19,20,21,13,19,18,20,7,18,8,15,17,16,13,10,4,9,
  7,8,10,9,11,9,11,10,12,12,5,13,4,6,6,13,8,9,10,13,13,
  11,10,5,3,3,4,9,6,8,3,5,3,2,2,1,3,5,11,2,3,5,6,9,8,5,
  2,5,3,4,6,4,8,15,12,16,20,18,23,18,19,24,23,24,21,26,
  36,38,37,39,45,42,41,37,38,38,35,37,35,31,32,30,20,39,
```

```
  40,33,32,35,34,36,34,32,33,27,28,25,22,17,18,16,10,9,
  5,12,7,8,8,9,19,21,24,20,23,19,17,18,17,22,11,12,3,9,
  10,4,5,13,3,5,6,3,5,4,2,5,1,2,4,4,3,2,1)

# Assummptions (probability vector, transition matrix,
# and distribution parameters)

delta <- c(0.25,0.25,0.25,0.25)
gamma <- 0.7 * diag(length(delta)) + rep(0.3 / length(delta))
distribution_class <- "pois"
distribution_theta <- list(lambda = c(4,9,17,25))

# log-likelihood
logL <- forward_backward_algorithm (x = x,
                                    delta = delta, gamma=gamma,
                                    distribution_class= distribution_class,
                                    distribution_theta=distribution_theta)$logL

# the Poisson distribution has one paramter, hence k=1
AIC_HMM(logL = logL, m = length(delta), k = 1)
BIC_HMM(size = length(x) , logL = logL, m = length(delta), k = 1)
```

`Baum_Welch_algorithm` *Estimation Using the Baum-Welch Algorithm*

Description

Estimates the parameters of a (non-stationary) discrete-time hidden Markov model. The Baum-Welch algorithm is a version of the EM (Estimation/Maximization) algorithm. See MacDonald & Zucchini (2009, Paragraph 4.2) for further details.

Usage

```
Baum_Welch_algorithm(
  x,
  m,
  delta,
  gamma,
  distribution_class,
  distribution_theta,
  discr_logL = FALSE,
  discr_logL_eps = 0.5,
  BW_max_iter = 50,
  BW_limit_accuracy = 0.001,
  BW_print = TRUE,
  Mstep_numerical = FALSE,
  DNM_limit_accuracy = 0.001,
  DNM_max_iter = 50,
```

```
  DNM_print = 2
)
```

Arguments

x	a vector object containing the time-series of observations that are assumed to be realizations of the (hidden Markov state dependent) observation process of the model.
m	integer; a (finite) number of states in the hidden Markov chain.
delta	vector object containing starting values for the marginal probability distribution of the m states of the Markov chain at the time point t=1 for the Baum-Welch algorithm.
gamma	a matrix (ncol=nrow=m) containing starting values for the transition matrix of the hidden Markov chain
distribution_class	a single character string object with the abbreviated name of the m observation distributions of the Markov dependent observation process. The following distributions are supported: Poisson (pois); generalized Poisson (genpois, parameter estimation via the Baum-Welch algorithm is only supported if the M-step is performed numerically, i.e. if Mstep_numerical = TRUE); normal (norm)
distribution_theta	a list object containing starting values for the parameters of the m observation distributions of the observation process that are dependent on the hidden Markov state.
discr_logL	a logical object indicating whether the discrete log-likelihood should be used (for distribution_class="norm") for estimating the model specific parameters instead of the general log-likelihood. See MacDonald & Zucchini (2009, Paragraph 1.2.3) for further details. Default value is FALSE.
discr_logL_eps	a single numerical value to approximately determine the discrete likelihood for a hidden Markov model based on nomal distributions (for "norm"). Default value is 0.5. See MacDonald & Zucchini (2009, Paragraph 1.2.3) for further details.
BW_max_iter	a single numerical value representing the maximum number of iterations in the Baum-Welch algorithm. Default value is 50.
BW_limit_accuracy	a single numerical value representing the convergence criterion of the Baum-Welch algorithm. Default value is 0.001.
BW_print	a logical object indicating whether the log-likelihood at each iteration-step shall be printed. Default value is TRUE.
Mstep_numerical	a logical object indicating whether the Maximization Step of the Baum-Welch algorithm shall be performed by numerical maximization using the nlm-function. Default value is FALSE.
DNM_limit_accuracy	a single numerical value representing the convergence criterion of the numerical maximization algorithm using the nlm-function (used to perform the M-step of the Baum-Welch-algorithm). Default value is 0.001.

`DNM_max_iter` a single numerical value representing the maximum number of iterations of the numerical maximization using the nlm-function (used to perform the M-step of the Baum-Welch-algorithm). Default value is `50`.

`DNM_print` a single numerical value to determine the level of printing of the `nlm`-function. See `nlm`-function for further informations. The value `0` suppresses, that no printing will be outputted. Default value is `2` for full printing.

Value

`Baum_Welch_algorithm` returns a list containing the estimated parameters of the hidden Markov model and other components. See MacDonald & Zucchini (2009, Paragraph 4.2) for further details on the calculated objects within this algorithm.

x input time-series of observations.

m input number of hidden states in the Markov chain.

zeta a (T,m)-matrix (when T indicates the length/size of the observation time-series and m the number of states of the HMM) containing probabilities (estimates of the conditional expectations of the missing data given the observations and the estimated model specific parameters) calculated by the algorithm. See MacDonald & Zucchini (2009, Paragraph 4.2.2) for further details.

eta a (T,m,m)-dimensional-array (when T indicates the length of the observation time-series and m the number of states of the HMM) containing probabilities (estimates of the conditional expectations of the missing data given the observations and the estimated model specific parameters) calculated by the algorithm. See MacDonald & Zucchini (2009, Paragraph 4.2.2) for further details.

logL a numerical value representing the logarithmized likelihood calculated by the `forward_backward_algorithm`.

iter number of performed iterations.

BIC a numerical value representing the Bayesian information criterion for the hidden Markov model with estimated parameters.

delta a vector object containing the estimates for the marginal probability distribution of the `m` states of the Markov chain at time-point point `t=1`.

gamma a matrix containing the estimates for the transition matrix of the hidden Markov chain.

... other input values (as arguments above). In the case that the algorithm stops before the targeted accuracy or the maximum number of iterations has been reached, further values are displayed and the estimates from the last successful iteration step are saved.

Author(s)

The basic algorithm for a Poisson-HMM is provided by MacDonald & Zucchini (2009, Paragraph 4.2, Paragraph A.2.3). Extension and implementation by Vitali Witowski (2013).

References

Baum, L., Petrie, T., Soules, G., Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. The annals of mathematical statistics, vol. **41**(1), 164–171.

Dempster, A., Laird, N., Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society. Series B (Methodological), vol. **39**(1), 1–38.

MacDonald, I. L., Zucchini, W. (2009) *Hidden Markov Models for Time Series: An Introduction Using R*, Boca Raton: Chapman & Hall.

See Also

HMM_based_method, HMM_training, direct_numerical_maximization, forward_backward_algorithm, initial_parameter_training

Examples

```
x <- c(1,16,19,34,22,6,3,5,6,3,4,1,4,3,5,7,9,8,11,11,
  14,16,13,11,11,10,12,19,23,25,24,23,20,21,22,22,18,7,
  5,3,4,3,2,3,4,5,4,2,1,3,4,5,4,5,3,5,6,4,3,6,4,8,9,12,
  9,14,17,15,25,23,25,35,29,36,34,36,29,41,42,39,40,43,
  37,36,20,20,21,22,23,26,27,28,25,28,24,21,25,21,20,21,
  11,18,19,20,21,13,19,18,20,7,18,8,15,17,16,13,10,4,9,
  7,8,10,9,11,9,11,10,12,12,5,13,4,6,6,13,8,9,10,13,13,
  11,10,5,3,3,4,9,6,8,3,5,3,2,2,1,3,5,11,2,3,5,6,9,8,5,
  2,5,3,4,6,4,8,15,12,16,20,18,23,18,19,24,23,24,21,26,
  36,38,37,39,45,42,41,37,38,38,35,37,35,31,32,30,20,39,
  40,33,32,35,34,36,34,32,33,27,28,25,22,17,18,16,10,9,
  5,12,7,8,8,9,19,21,24,20,23,19,17,18,17,22,11,12,3,9,
  10,4,5,13,3,5,6,3,5,4,2,5,1,2,4,4,3,2,1)

  # Assumptions (number of states, probability vector,
  # transition matrix, and distribution parameters)

  m <- 4
  delta <- c(0.25,0.25,0.25,0.25)
  gamma <- 0.7 * diag(m) + rep(0.3 / m)
  distribution_class <- "pois"
  distribution_theta <- list(lambda = c(4,9,17,25))

  # Estimation of a HMM using the Baum-Welch algorithm

    trained_HMM_with_m_hidden_states <-
     Baum_Welch_algorithm(x = x,
                          m = m,
                      delta = delta,
                      gamma = gamma,
          distribution_class = distribution_class,
          distribution_theta = distribution_theta)

 print(trained_HMM_with_m_hidden_states)
```

BIC_HMM *BIC Value for a Discrete Time Hidden Markov Model*

Description

Computes the Aikaike's information criterion and the Bayesian information criterion for a discrete time hidden Markov model, given a time-series of observations.

Usage

```
BIC_HMM(size, m, k, logL)
```

Arguments

size	length of the time-series of observations x (also T)
m	integer number of states in the Markov chain of the model
k	single integer value representing the number of parameters of the underlying distribution of the observation process (e.g. k=2 for the normal distribution (mean and standard deviation))
logL	logarithmized likelihood of the model

Details

For a discrete-time hidden Markov model, AIC and BIC are as follows (MacDonald & Zucchini (2009, Paragraph 6.1 and A.2.3)):

$$\text{AIC} = -2logL + 2p$$

$$\text{BIC} = -2logL + p \log T$$

where T indicates the length/size of the observation time-series and p denotes the number of independent parameters of the model. In case of a HMM as provided by this package, where k and m are defined as in the arguments above, p can be calculated as follows:

$$p = m^2 + km - 1.$$

Value

The AIC or BIC value of the fitted hidden Markov model.

Author(s)

Based on MacDonald & Zucchini (2009, Paragraph 6.1 and A.2.3). Implementation by Vitali Witowski (2013).

References

MacDonald, I. L., Zucchini, W. (2009) *Hidden Markov Models for Time Series: An Introduction Using R*, Boca Raton: Chapman & Hall.

See Also

AIC_HMM, HMM_training

Examples

```
x <- c(1,16,19,34,22,6,3,5,6,3,4,1,4,3,5,7,9,8,11,11,
  14,16,13,11,11,10,12,19,23,25,24,23,20,21,22,22,18,7,
  5,3,4,3,2,3,4,5,4,2,1,3,4,5,4,5,3,5,6,4,3,6,4,8,9,12,
  9,14,17,15,25,23,25,35,29,36,34,36,29,41,42,39,40,43,
  37,36,20,20,21,22,23,26,27,28,25,28,24,21,25,21,20,21,
  11,18,19,20,21,13,19,18,20,7,18,8,15,17,16,13,10,4,9,
  7,8,10,9,11,9,11,10,12,12,5,13,4,6,6,13,8,9,10,13,13,
  11,10,5,3,3,4,9,6,8,3,5,3,2,2,1,3,5,11,2,3,5,6,9,8,5,
  2,5,3,4,6,4,8,15,12,16,20,18,23,18,19,24,23,24,21,26,
  36,38,37,39,45,42,41,37,38,38,35,37,35,31,32,30,20,39,
  40,33,32,35,34,36,34,32,33,27,28,25,22,17,18,16,10,9,
  5,12,7,8,8,9,19,21,24,20,23,19,17,18,17,22,11,12,3,9,
  10,4,5,13,3,5,6,3,5,4,2,5,1,2,4,4,3,2,1)

# Assummptions (probability vector, transition matrix,
# and distribution parameters)

delta <- c(0.25,0.25,0.25,0.25)
gamma <- 0.7 * diag(length(delta)) + rep(0.3 / length(delta))
distribution_class <- "pois"
distribution_theta <- list(lambda = c(4,9,17,25))

# log-likelihood
logL <- forward_backward_algorithm (x = x,
                                    delta = delta, gamma=gamma,
                                    distribution_class= distribution_class,
                                    distribution_theta=distribution_theta)$logL

# the Poisson distribution has one paramter, hence k=1
AIC_HMM(logL = logL, m = length(delta), k = 1)
BIC_HMM(size = length(x) , logL = logL, m = length(delta), k = 1)
```

---

cut_off_point_method *Cut-Off Point Method for Assigning Physical Activity Patterns*

Description

This function assigns an activity range to each observation of a time-series, such as for a sequence of impulse counts recorded by an accelerometer. The activity ranges are defined by thresholds called "cut-off points". Furthermore, bout periods are analysed (see Details for further informations).

Usage

```
cut_off_point_method(
  x,
  cut_points,
  names_activity_ranges = NA,
  hidden_PA_levels = NA,
  bout_lengths = NULL,
  plotting = 0
)
```

Arguments

x	a vector object of length T containing non-negative observations of a time-series, such as a sequence of accelerometer impulse counts.
cut_points	a vector object containing cut-off points to separate activity ranges. For instance, the vector c(7,15,23) separates the four activity ranges [0,7);[7,15);[15,23);[23,Inf).
names_activity_ranges	an optional character string vector to name the activity ranges induced by the cut-points. This vector must contain one element more than the vector cut_points.
hidden_PA_levels	an optional vector object of length T containing a sequence of the estimated hidden physical activity levels (i.e. means) underlying the time-series of accelerometer counts. Such a sequence can be extracted by decoding a trained hidden Markov model. The cut-point method classifies then each count by its level in the hidden Markov chain that generates the physical activity counts, and does not use the observed count value (see HMM_based_method for further details). Default is NA (for the traditional cut-point method).
bout_lengths	a vector object (with even number of elemets) to define the range of the bout intervals (see Details for the definition of bouts). For instance, bout_lengths = c(1,1,2,2,3,10,11,20,1,20) defines the five bout intervals [1,1] (1 count); [2,2] (2 counts); [3,10] (3-10 counts); [11,20] (11-20 counts); [1,20] (1-20 counts - overlapping with other bout intervalls is possible). Default value is bout_lengths=NULL.
plotting	a numeric value between 0 and 5 (generates different outputs). NA suppresses graphical output. Default value is 0. 0: output 1-5 1: summary of all results 2: time series of activity counts, classified into activity ranges 3: time series of bouts (and, if available, the sequence of the estimated hidden physical activity levels, extracted by decoding a trained HMM, in green color) 4: barplots of absolute and relative frequencies of time spent in different activity ranges 5: barplots of absolute frequencies of different bout intervals (overall and by activity ranges)

Details

A bout is defined as a period of time spending a defined intensity of physical activities in a specified physical activity range, without switching to activity intensities in a different activity range.

Value

`cut_off_point_method` returns a list containing the extracted sequence of activity ranges and plots key figures.

activity_ranges an array object containing the cut-off intervals that indicate the activity ranges.

classification an integer vector containing the sequence of activity ranges that were assigned to the observed time-series of accelerometer counts. If `hidden_PA_levels=NA`, then `classification` is the output of the traditional cut-point method, meaning that an activity range has been assigned to each accelerometer count over its observed value actual position. In case when `hidden_PA_levels` is available, `classification` is the output of the extendend cut-point method using hidden Markov models (see `HMM_based_method` for further details).

classification_per_activity_range a pairlist object containing the classification of the observed counts by the assigned activity range.

freq_acitvity_range table object containing the absolute frequencies of classifications into activity ranges.

rel_freq_acitvity_range table object containing the relative frequencies of classifications into activity ranges.

quantity_of_bouts overall number of bouts.

bout_periods an array including the bout length assigned to acitiy ranges.

abs_freq_bouts_el a pairlist object containing the absolute frequency of bout length per epoch length (aggregated).

Author(s)

Vitali Witowski (2013)

See Also

HMM_based_method

Examples

```
x <- c(1,16,19,34,22,6,3,5,6,3,4,1,4,3,5,7,9,8,11,11,
  14,16,13,11,11,10,12,19,23,25,24,23,20,21,22,22,18,7,
  5,3,4,3,2,3,4,5,4,2,1,3,4,5,4,5,3,5,6,4,3,6,4,8,9,12,
  9,14,17,15,25,23,25,35,29,36,34,36,29,41,42,39,40,43,
  37,36,20,20,21,22,23,26,27,28,25,28,24,21,25,21,20,21,
  11,18,19,20,21,13,19,18,20,7,18,8,15,17,16,13,10,4,9,
  7,8,10,9,11,9,11,10,12,12,5,13,4,6,6,13,8,9,10,13,13,
  11,10,5,3,3,4,9,6,8,3,5,3,2,2,1,3,5,11,2,3,5,6,9,8,5,
  2,5,3,4,6,4,8,15,12,16,20,18,23,18,19,24,23,24,21,26,
  36,38,37,39,45,42,41,37,38,38,35,37,35,31,32,30,20,39,
  40,33,32,35,34,36,34,32,33,27,28,25,22,17,18,16,10,9,
```

```
  5,12,7,8,8,9,19,21,24,20,23,19,17,18,17,22,11,12,3,9,
  10,4,5,13,3,5,6,3,5,4,2,5,1,2,4,4,3,2,1)

# 1) Traditional cut point method -----------------------
# Assigning activity ranges to activity counts using
# fictitious cut-off points that produce the four activity
# ranges "sedentary"", "light"", "moderate"", and "vigorous".

  solution_of_traditional_cut_off_point_method <-
      cut_off_point_method(x = x,
                           cut_points = c(5,15,23),
                           names_activity_ranges = c("SED","LIG","MOD","VIG"),
                          bout_lengths = c(1,1,2,2,3,3,4,4,5,5,6,12,13,40,41,265,1,265),
                           plotting = 0)
 print(solution_of_traditional_cut_off_point_method)


# 2) Extension of the traditional cut_point method
#     using HMMs
# The following three steps define an extension of the
# traditional cut-off method by first extracting the hidden
# physical activity pattern behind the accelerometer counts
# using a HMM (those three steps are basically combined in
# the function HMM_based_method, see HMM_based_method for
# further details and references):

# Step 1 ---
# Train hidden Markov model for different number of
# states m=2,...,6 and select the model with the most
# plausible m

m_trained_HMM <-
  HMM_training(x = x,
               min_m = 2,
               max_m = 6, BW_print=FALSE,
               distribution_class = "pois")$trained_HMM_with_selected_m


 # Step 2 ---
 # Decode the trained HMM (by using the
 # Viterbi algorithm (global decoding)) to get the estimated
 # sequence of hidden physical activity levels
 # underlying the the accelerometer counts

 # You have to compute 'm_trained_HMM' first (see Step 1)

   global_decoding <-
     HMM_decoding(x = x,
                  m = m_trained_HMM$m,
                  delta = m_trained_HMM$delta,
                  gamma = m_trained_HMM$gamma,
                  distribution_class = m_trained_HMM$distribution_class,
                  distribution_theta = m_trained_HMM$distribution_theta,
```

```
                decoding_method = "global")
hidden_PA_levels <- global_decoding$decoding_distr_means


# Step 3 ---
# Assigning activity ranges to activity counts using the
# information extracted by decoding the HMM for the counts
# (PA-levels) and fictitious cut-off points that produce
# four so-called activity ranges:"sedentary", "light",
# "moderate" and "vigorous":

# You have to compute 'm_trained_HMM' and 'hidden_PA_levels' first (see above)

  solution_of_HMM_based_cut_off_point_method <-
      cut_off_point_method(x = x,
                           hidden_PA_levels = hidden_PA_levels,
                           cut_points = c(5,15,23),
                           names_activity_ranges = c("SED","LIG","MOD","VIG"),
                         bout_lengths = c(1,1,2,2,3,3,4,4,5,5,6,12,13,40,41,265,1,265),
                           plotting=1)
```

---

dgenpois *The Generalized Poisson Distribution*

Description

Density function for the generalized Poisson distribution.

Usage

```
dgenpois(x, lambda1, lambda2)
```

Arguments

x	a vector of (non-negative integer) quantiles
lambda1	a single numeric value for parameter lambda1 with $lambda1 > 0$
lambda2	a single numeric value for parameter lambda2 with $0 \leq lamdba2 < 1$. When lambda2=0, the generalized Poisson distribution reduces to the Poisson distribution

Details

The generalized Poisson distribution has the density

$$p(x) = \lambda_1(\lambda_1 + \lambda_2 \cdot x)^{x-1} \frac{\exp(-\lambda_1 - \lambda_2 \cdot x))}{x!}$$

for $x = 0, 1, 2, \ldots$,b with $\mathrm{E}(X) = \frac{\lambda_1}{1-\lambda_2}$ and variance $\mathrm{var}(X) = \frac{\lambda_1}{(1-\lambda_2)^3}$.

Value

dgenpois gives the density of the generalized Poisson distribution.

Author(s)

Based on Joe and Zhu (2005). Implementation by Vitali Witowski (2013).

References

Joe, H., Zhu, R. (2005). Generalized poisson distribution: the property of mixture of poisson and comparison with negative binomial distribution. Biometrical Journal **47**(2):219–229.

See Also

pgenpois, rgenpois; Distributions for other standard distributions, including dpois for the Poisson distribution.

Examples

```
dgenpois(x = seq(0,20), lambda1 = 10, lambda2 = 0.5)
pgenpois(q = 5, lambda1 = 10, lambda2 = 0.5)
hist(rgenpois(n = 1000, lambda1 = 10, lambda2 = 0.5) )
```

---

direct_numerical_maximization

Estimation by Directly Maximizing the log-Likelihood

Description

Estimates the parameters of a (stationary) discrete-time hidden Markov model by directly maximizing the log-likelihood of the model using the nlm-function. See MacDonald & Zucchini (2009, Paragraph 3) for further details.

Usage

```
direct_numerical_maximization(
  x,
  m,
  delta,
  gamma,
  distribution_class,
  distribution_theta,
  DNM_limit_accuracy = 0.001,
  DNM_max_iter = 50,
  DNM_print = 2
)
```

Arguments

x	a vector object containing the time-series of observations that are assumed to be realizations of the (hidden Markov state dependent) observation process of the model.
m	interger ;a (finite) number of states in the hidden Markov chain
delta	a vector object containing starting values for the marginal probability distribution of the m states of the Markov chain at the time point t=1. This implementation of the algorithm uses the stationary distribution as delta.
gamma	a matrix (nrow=ncol=m) containing starting values for the transition matrix of the hidden Markov chain
distribution_class	a single character string object with the abbreviated name of the m observation distributions of the Markov dependent observation process. The following distributions are supported by this algorithm: Poisson (pois); generalized Poisson (genpois); normal (norm, discrete log-Likelihood not applicable by this algorithm)
distribution_theta	a list object containing starting values for the parameters of the m observation distributions that are dependent on the hidden Markov state
DNM_limit_accuracy	a single numerical value representing the convergence criterion of the direct numerical maximization algorithm using the nlm-function. Default value is 0.001.
DNM_max_iter	a single numerical value representing the maximum number of iterations of the direct numerical maximization using the nlm-function. Default value is 50.
DNM_print	a single numerical value to determine the level of printing of the nlm-function. See nlm-function for further informations. The value 0 suppresses, that no printing will be outputted. Default value is 2 for full printing.

Value

direct_numerical_maximization returns a list containing the estimated parameters of the hidden Markov model and other components.

x input time-series of observations.

m input number of hidden states in the Markov chain.

logL a numerical value representing the logarithmized likelihood calculated by the forward_backward_algorithm.

AIC a numerical value representing Akaike's information criterion for the hidden Markov model with estimated parameters.

BIC a numerical value representing the Bayesian information criterion for the hidden Markov model with estimated parameters.

delta a vector object containing the estimates for the marginal probability distribution of the m states of the Markov chain at time-point point t=1.

gamma a matrix containing the estimates for the transition matrix of the hidden Markov chain.

distribution_theta a list object containing estimates for the parameters of the m observation distributions that are dependent on the hidden Markov state.

distribution_class input distribution class.

Author(s)

The basic algorithm of a Poisson-HMM is provided by MacDonald & Zucchini (2009, Paragraph A.1). Extension and implementation by Vitali Witowski (2013).

References

MacDonald, I. L., Zucchini, W. (2009) *Hidden Markov Models for Time Series: An Introduction Using R*, Boca Raton: Chapman & Hall.

See Also

HMM_based_method, HMM_training, Baum_Welch_algorithm, forward_backward_algorithm, initial_parameter_trai

Examples

```
x <- c(1,16,19,34,22,6,3,5,6,3,4,1,4,3,5,7,9,8,11,11,
  14,16,13,11,11,10,12,19,23,25,24,23,20,21,22,22,18,7,
  5,3,4,3,2,3,4,5,4,2,1,3,4,5,4,5,3,5,6,4,3,6,4,8,9,12,
  9,14,17,15,25,23,25,35,29,36,34,36,29,41,42,39,40,43,
  37,36,20,20,21,22,23,26,27,28,25,28,24,21,25,21,20,21,
  11,18,19,20,21,13,19,18,20,7,18,8,15,17,16,13,10,4,9,
  7,8,10,9,11,9,11,10,12,12,5,13,4,6,6,13,8,9,10,13,13,
  11,10,5,3,3,4,9,6,8,3,5,3,2,2,1,3,5,11,2,3,5,6,9,8,5,
  2,5,3,4,6,4,8,15,12,16,20,18,23,18,19,24,23,24,21,26,
  36,38,37,39,45,42,41,37,38,38,35,37,35,31,32,30,20,39,
  40,33,32,35,34,36,34,32,33,27,28,25,22,17,18,16,10,9,
  5,12,7,8,8,9,19,21,24,20,23,19,17,18,17,22,11,12,3,9,
  10,4,5,13,3,5,6,3,5,4,2,5,1,2,4,4,3,2,1)

  # Assumptions (number of states, probability vector,
  # transition matrix, and distribution parameters)

  m <- 4
  delta <- c(0.25,0.25,0.25,0.25)
  gamma <- 0.7 * diag(m) + rep(0.3 / m)
  distribution_class <- "pois"
  distribution_theta <- list(lambda = c(4,9,17,25))

  # Estimation of a HMM using the method of
  # direct numerical maximization

    trained_HMM_with_m_hidden_states <-
      direct_numerical_maximization(x = x,
                                    m = m,
                                delta = delta,
                                gamma = gamma,
                  distribution_class = distribution_class,
                        DNM_max_iter = 100,
                  distribution_theta = distribution_theta)

    print(trained_HMM_with_m_hidden_states)
```

`forward_backward_algorithm` *Calculating Forward and Backward Probabilities and Likelihood*

Description

The function calculates the logarithmized forward and backward probabilities and the logarithmized likelihood for a discrete time hidden Markov model, as defined in MacDonald & Zucchini (2009, Paragraph 3.1- Paragraph 3.3 and Paragraph 4.1).

Usage

```
forward_backward_algorithm(
  x,
  delta,
  gamma,
  distribution_class,
  distribution_theta,
  discr_logL = FALSE,
  discr_logL_eps = 0.5
)
```

Arguments

`x`	a vector object containing the time-series of observations that are assumed to be realizations of the (hidden Markov state dependent) observation process of the model.
`delta`	a vector object containing values for the marginal probability distribution of the `m` states of the Markov chain at the time point `t=1`.
`gamma`	a matrix (`nrow=ncol=m`) containing values for the transition matrix of the hidden Markov chain.
`distribution_class`	a single character string object with the abbreviated name of the `m` observation distributions of the Markov dependent observation process. The following distributions are supported: Poisson (`pois`); generalized Poisson (`genpois`); normal (`norm`); geometric (`geom`).
`distribution_theta`	a list object containing the parameter values for the `m` observation distributions of the observation process that are dependent on the hidden Markov state.
`discr_logL`	a logical object. It is `TRUE` if the discrete log-likelihood shall be calculated (for `distribution_class="norm"`) instead of the general log-likelihood. See MacDonald & Zucchini (2009, Paragraph 1.2.3) for further details. Default is `FALSE`.
`discr_logL_eps`	a single numerical value to approximately determine the discrete log-likelihood for a hidden Markov model based on normal distributions (for `"norm"`). The default value is `0.5`. See MacDonald & Zucchini (2009, Paragraph 1.2.3) for further details.

Value

`forward_backward_algorithm` returns a list containing the logarithmized forward and backward probabilities and the logarithmized likelihood.

log_alpha a (T,m)-matrix (when T indicates the length/size of the observation time-series and m the number of states of the HMM) containing the logarithmized forward probabilities.

log_beta a (T,m)-matrix (when T indicates the length/size of the observation time-series and m the number of states of the HMM) containing the logarithmized backward probabilities.

logL a single numerical value representing the logarithmized likelihood.

logL_calculation a single character string object which indicates how `logL` has been calculated (see Zucchini (2009) Paragraph 3.1-3.4, 4.1, A.2.2, A.2.3 for further details).

Author(s)

The basic algorithm for a Poisson-HMM is provided by MacDonald & Zucchini (2009, Paragraph A.2.2). Extension and implementation by Vitali Witowski (2013).

References

MacDonald, I. L., Zucchini, W. (2009) *Hidden Markov Models for Time Series: An Introduction Using R*, Boca Raton: Chapman & Hall.

See Also

`HMM_based_method`, `HMM_training`, `Baum_Welch_algorithm`, `direct_numerical_maximization`, `initial_parameter_training`

Examples

```
x <- c(1,16,19,34,22,6,3,5,6,3,4,1,4,3,5,7,9,8,11,11,
  14,16,13,11,11,10,12,19,23,25,24,23,20,21,22,22,18,7,
  5,3,4,3,2,3,4,5,4,2,1,3,4,5,4,5,3,5,6,4,3,6,4,8,9,12,
  9,14,17,15,25,23,25,35,29,36,34,36,29,41,42,39,40,43,
  37,36,20,20,21,22,23,26,27,28,25,28,24,21,25,21,20,21,
  11,18,19,20,21,13,19,18,20,7,18,8,15,17,16,13,10,4,9,
  7,8,10,9,11,9,11,10,12,12,5,13,4,6,6,13,8,9,10,13,13,
  11,10,5,3,3,4,9,6,8,3,5,3,2,2,1,3,5,11,2,3,5,6,9,8,5,
  2,5,3,4,6,4,8,15,12,16,20,18,23,18,19,24,23,24,21,26,
  36,38,37,39,45,42,41,37,38,38,35,37,35,31,32,30,20,39,
  40,33,32,35,34,36,34,32,33,27,28,25,22,17,18,16,10,9,
  5,12,7,8,8,9,19,21,24,20,23,19,17,18,17,22,11,12,3,9,
  10,4,5,13,3,5,6,3,5,4,2,5,1,2,4,4,3,2,1)

  # Assumptions (number of states, probability vector,
  # transition matrix, and distribution parameters)

  m <- 4
  delta <- c(0.25,0.25,0.25,0.25)
  gamma <- 0.7 * diag(m) + rep(0.3 / m)
  distribution_class <- "pois"
```

```
  distribution_theta <- list(lambda = c(4,9,17,25))

# Calculating logarithmized forward/backward probabilities
# and logarithmized likelihood


forward_and_backward_probabilities_and_logL <-
forward_backward_algorithm (x = x,
                        delta = delta,
                        gamma = gamma,
           distribution_class = distribution_class,
           distribution_theta = distribution_theta)

 print(forward_and_backward_probabilities_and_logL)
```

HMM_based_method *Hidden Markov Method for Predicting Physical Activity Patterns*

Description

This function assigns a physical activity range to each observation of a time-series (such as a sequence of impulse counts recorded by an accelerometer) using hidden Markov models (HMM). The activity ranges are defined by thresholds called cut-off points. Basically, this function combines HMM_training, HMM_decoding and cut_off_point_method. See Details for further information.

Usage

```
HMM_based_method(
  x,
  cut_points,
  distribution_class,
  min_m = 2,
  max_m = 6,
  n = 100,
  max_scaled_x = NA,
  names_activity_ranges = NA,
  discr_logL = FALSE,
  discr_logL_eps = 0.5,
  dynamical_selection = TRUE,
  training_method = "EM",
  Mstep_numerical = FALSE,
  BW_max_iter = 50,
  BW_limit_accuracy = 0.001,
  BW_print = TRUE,
  DNM_max_iter = 50,
  DNM_limit_accuracy = 0.001,
  DNM_print = 2,
```

```
  decoding_method = "global",
  bout_lengths = NULL,
  plotting = 0
)
```

Arguments

x	a vector object of length T containing non-negative observations of a time-series, such as a sequence of accelerometer impulse counts, which are assumed to be realizations of the (hidden Markov state dependent) observation process of a HMM.
cut_points	a vector object containing cut-off points to separate activity ranges. For instance, the vector c(7,15,23) separates the four activity ranges [0,7), [7,15), [15,23) and [23,Inf).
distribution_class	a single character string object with the abbreviated name of the m observation distributions of the Markov dependent observation process. The following distributions are supported: Poisson (pois); generalized Poisson (genpois); normal (norm)).
min_m	miminum number of hidden states in the hidden Markov chain. Default value is 2.
max_m	maximum number of hidden states in the hidden Markov chain. Default value is 6.
n	a single numerical value specifying the number of samples. Default value is 100.
max_scaled_x	an optional numerical value, to be used to scale the observations of the time-series x before the hidden Markov model is trained and decoded (see Details). Default value is NA.
names_activity_ranges	an optional character string vector to name the activity ranges induced by the cut-points. This vector must contain one element more than the vector cut_points.
discr_logL	a logical object indicating whether the discrete log-likelihood should be used (for "norm") for estimating the model specific parameters instead of the general log-likelihood. See MacDonald & Zucchini (2009, Paragraph 1.2.3) for further details. Default is FALSE.
discr_logL_eps	a single numerical value to approximate the discrete log-likelihood for a hidden Markov model based on nomal distributions (for distribution_class="norm"). The default value is 0.5.
dynamical_selection	a logical value indicating whether the method of dynamical initial parameter selection should be applied (see HMM_training for details). Default is TRUE.
training_method	a logical value indicating whether the Baum-Welch algorithm ("EM") or the method of direct numerical maximization ("numerical") should be applied for estimating the model specific parameters of the HMM. See Baum_Welch_algorithm and direct_numerical_maximization for further details. Default is training_method = "EM".

`Mstep_numerical`
: a logical object indicating whether the Maximization Step of the Baum-Welch algorithm shall be performed by numerical maximization. Default is FALSE.

`BW_max_iter`
: a single numerical value representing the maximum number of iterations in the Baum-Welch algorithm. Default value is `50`.

`BW_limit_accuracy`
: a single numerical value representing the convergence criterion of the Baum-Welch algorithm. Default value is `0.001`.

`BW_print`
: a logical object indicating whether the log-likelihood at each iteration-step shall be printed. Default is `TRUE`.

`DNM_max_iter`
: a single numerical value representing the maximum number of iterations of the numerical maximization using the nlm-function (used to perform the M-step of the Baum-Welch-algorithm). Default value is `50`.

`DNM_limit_accuracy`
: a single numerical value representing the convergence criterion of the numerical maximization algorithm using the nlm function (used to perform the M-step of the Baum-Welch-algorithm). Default value is `0.001`.

`DNM_print`
: a single numerical value to determine the level of printing of the `nlm`-function. See `nlm`-function for further informations. The value `0` suppresses, that no printing will be outputted. Default value is `2` for full printing.

`decoding_method`
: a string object to choose the applied decoding-method to decode the HMM given the time-series of observations `x`. Possible values are `"global"` (for the use of the `Viterbi_algorithm`) and `"local"` (for the use of the `local_decoding_algorithm`). Default value is `"global"`.

`bout_lengths`
: a vector object (with even number of elemets) to define the range of the bout intervals (see Details for the definition of bouts). For instance, `bout_lengths = c(1,1,2,2,3,10,11,20,1,20)` defines the five bout intervals [1,1] (1 count); [2,2] (2 counts); [3,10] (3-10 counts); [11,20] (11-20 counts); [1,20] (1-20 counts - overlapping with other bout intervalls is possible). Default value is `bout_lengths=NULL`.

`plotting`
: a numeric value between 0 and 5 (generates different outputs). NA suppresses graphical output. Default value is `0`.
`0`: output 1-5
`1`: summary of all results
`2`: time series of activity counts, classified into activity ranges
`3`: time series of bouts (and, if available, the sequence of the estimated hidden physical activity levels, extracted by decoding a trained HMM, in green colour)
`4`: barplots of absolute and relative frequencies of time spent in different activity ranges
`5`: barplots of relative frequencies of the lenghts of bout intervals (overall and by activity ranges)

Details

The function combines `HMM_training`, `HMM_decoding` and `cut_off_point_method` as follows:

Step 1: HMM_training trains the most likely HMM for a given time-series of accelerometer counts.
Step 2: HMM_decoding decodes the trained HMM (Step 1) into the most likely sequence of hidden states corresponding to the given time-series of observations (respectively the most likely sequence of physical activity levels corresponding to the time-series of accelerometer counts).
Step 3. cut_off_point_method assigns an activity range to each accelerometer count by its hidden physical activity level (extracted in Step 2).

Value

HMM_based_method returns a list containing the output of the trained hidden Markov model, including the selected number of states m (i.e., number of physical activities) and plots key figures.

trained_HMM_with_selected_m a list object containing the trained hidden Markov model including the selected number of states m (see HMM_training for further details).

decoding a list object containing the output of the decoding (see HMM_decoding for further details).

extendend_cut_off_point_method a list object containing the output of the cut-off point method. The counts x are classified into the activity ranges by the corresponding sequence of hidden PA-levels, which were decoded by the HMM (see cut_off_point_method for further details).

Note

The parameter max_scaled_x can be applied to scale the values of the observations. This might prevent the alogrithm from numerical instabilities. At the end, the results are internaly rescaled to the original scale. For instance, a value of max_scaled_x=200 shrinks the count values of the complete time-series x to a maximum of 200. Training and decoding of the HMM is carried out using the scaled time-series.
From our experience, especially time-series with observations values >1500, or where T > 1000, show numerical instabilities. We then advice to make use of max_scaled_x .

The extention of the cut-off point method using a Poisson based HMM has been provided and evaluated successfully on simulated data firstly by Barbara Brachmann in her diploma thesis (see References).

Author(s)

Vitali Witowski (2013).

References

Brachmann, B. (2011). Hidden-Markov-Modelle fuer Akzelerometerdaten. Diploma Thesis, University Bremen - Bremen Institute for Prevention Research and Social Medicine (BIPS).

MacDonald, I. L., Zucchini, W. (2009) *Hidden Markov Models for Time Series: An Introduction Using R*, Boca Raton: Chapman & Hall.

Witowski, V., Foraita, R., Pitsiladis, Y., Pigeot, I., Wirsik, N. (2014) Using hidden Markov models to improve quantifying physical activity in accelerometer data - A simulation study. PLOS ONE. **9**(12), e114089. doi:10.1371/journal.pone.0114089

See Also

initial_parameter_training, Baum_Welch_algorithm, direct_numerical_maximization, AIC_HMM, BIC_HMM, HMM_training, Viterbi_algorithm, local_decoding_algorithm, cut_off_point_method

Examples

```
x <- c(1,16,19,34,22,6,3,5,6,3,4,1,4,3,5,7,9,8,11,11,
  14,16,13,11,11,10,12,19,23,25,24,23,20,21,22,22,18,7,
  5,3,4,3,2,3,4,5,4,2,1,3,4,5,4,5,3,5,6,4,3,6,4,8,9,12,
  9,14,17,15,25,23,25,35,29,36,34,36,29,41,42,39,40,43,
  37,36,20,20,21,22,23,26,27,28,25,28,24,21,25,21,20,21,
  11,18,19,20,21,13,19,18,20,7,18,8,15,17,16,13,10,4,9,
  7,8,10,9,11,9,11,10,12,12,5,13,4,6,6,13,8,9,10,13,13,
  11,10,5,3,3,4,9,6,8,3,5,3,2,2,1,3,5,11,2,3,5,6,9,8,5,
  2,5,3,4,6,4,8,15,12,16,20,18,23,18,19,24,23,24,21,26,
  36,38,37,39,45,42,41,37,38,38,35,37,35,31,32,30,20,39,
  40,33,32,35,34,36,34,32,33,27,28,25,22,17,18,16,10,9,
  5,12,7,8,8,9,19,21,24,20,23,19,17,18,17,22,11,12,3,9,
  10,4,5,13,3,5,6,3,5,4,2,5,1,2,4,4,3,2,1)

# Assumptions (number of states, probability vector,
# transition matrix, and distribution parameters)

m <- 4
delta <- c(0.25, 0.25, 0.25, 0.25)
gamma <- 0.7 * diag(m) + rep(0.3 / m)
distribution_class <- "pois"
distribution_theta <- list(lambda = c(4, 9, 17, 25))
```

HMM_decoding *Algorithm for Decoding Hidden Markov Models (local or global)*

Description

The function decodes a hidden Markov model into a most likely sequence of hidden states. Furthermore this function provides estimated observation values along the most likely sequence of hidden states. See Details for more information.

Usage

```
HMM_decoding(
  x,
  m,
  delta,
  gamma,
  distribution_class,
  distribution_theta,
  decoding_method = "global",
```

```
  discr_logL = FALSE,
  discr_logL_eps = 0.5
)
```

Arguments

x	a vector object containing the time-series of observations that are assumed to be realizations of the (hidden Markov state dependent) observation process of the model.
m	integer; (finite) number of states in the hidden Markov chain.
delta	a vector object containing values for the marginal probability distribution of the m states of the Markov chain at the time point t=1.
gamma	a matrix (ncol=nrow=m) containing values for the transition matrix of the hidden Markov chain.
distribution_class	a single character string object with the abbreviated name of the m observation distributions of the Markov dependent observation process. The following distributions are supported by this algorithm: Poisson (pois); generalized Poisson (genpois); normal (norm); geometric (geom).
distribution_theta	a list object containing the parameter values for the m observation distributions that are dependent on the hidden Markov state.
decoding_method	a string object to choose the applied decoding-method to decode the HMM given the time-series of observations x. Possible values are "global" (for the use of the Viterbi_algorithm) and "local" (for the use of the local_decoding_algorithm). Default value is "global".
discr_logL	a logical object. It is TRUE if the discrete log-likelihood shall be calculated (for distribution_class="norm" instead of the general log-likelihood). Default is FALSE.
discr_logL_eps	a single numerical value to approximately determine the discrete log-likelihood for a hidden Markov model based on nomal distributions (for "norm"). The default value is 0.5.

Details

More precisely, the function works as follows:

Step 1: In a first step, the algorithm decodes a HMM into the most likely sequence of hidden states, given a time-series of observations. The user can choose between a global and a local approch.
If decoding_method="global" is applied, the function calls Viterbi_algorithm to determine the sequence of most likely hidden states for all time points simultaneously.
If decoding_method="local" is applied, the function calls local_decoding_algorithm to determine the most likely hidden state for each time point seperately.

Step 2: In a second step, this function links each observation to the mean of the distribution, that corresponds to the decoded state at this point in time.

Value

`HMM_decoding` returns a list containing the following two components:

decoding_method a string object indicating the applied decoding method.

decoding a numerical vector containing the most likely sequence of hidden states as decoded by the `Viterbi_algorithm` (if `"global"` was applied) or by the `local_decoding_algorithm` (if `"local"` was applied).

decoding_distr_means a numerical vector of estimated oberservation values along the most likely seuquence of hidden states (see `decoding` and Step 2).

Author(s)

Vitali Witowski (2013).

References

MacDonald, I. L., Zucchini, W. (2009) *Hidden Markov Models for Time Series: An Introduction Using R*, Boca Raton: Chapman & Hall.

See Also

`local_decoding_algorithm`, `Viterbi_algorithm`

Examples

```
x <- c(1,16,19,34,22,6,3,5,6,3,4,1,4,3,5,7,9,8,11,11,
  14,16,13,11,11,10,12,19,23,25,24,23,20,21,22,22,18,7,
  5,3,4,3,2,3,4,5,4,2,1,3,4,5,4,5,3,5,6,4,3,6,4,8,9,12,
  9,14,17,15,25,23,25,35,29,36,34,36,29,41,42,39,40,43,
  37,36,20,20,21,22,23,26,27,28,25,28,24,21,25,21,20,21,
  11,18,19,20,21,13,19,18,20,7,18,8,15,17,16,13,10,4,9,
  7,8,10,9,11,9,11,10,12,12,5,13,4,6,6,13,8,9,10,13,13,
  11,10,5,3,3,4,9,6,8,3,5,3,2,2,1,3,5,11,2,3,5,6,9,8,5,
  2,5,3,4,6,4,8,15,12,16,20,18,23,18,19,24,23,24,21,26,
  36,38,37,39,45,42,41,37,38,38,35,37,35,31,32,30,20,39,
  40,33,32,35,34,36,34,32,33,27,28,25,22,17,18,16,10,9,
  5,12,7,8,8,9,19,21,24,20,23,19,17,18,17,22,11,12,3,9,
  10,4,5,13,3,5,6,3,5,4,2,5,1,2,4,4,3,2,1)

 # Set graphical parameters
 old.par <- par(no.readonly = TRUE)
 par(mfrow = c(1,1))

# i) Train hidden Markov model -----
#    for different number of states m=2,...,6 and select the optimal model

 m_trained_HMM <-
    HMM_training(x = x,
             min_m = 2,
             max_m = 6,
     distribution_class = "pois")$trained_HMM_with_selected_m
```

```
# ii) Global decoding -----
# Decode the trained HMM using the Viterbi algorithm to get
# the estimated sequence of hidden physical activity levels
global_decoding <-
    HMM_decoding(
        x = x,
        m = m_trained_HMM$m,
        delta = m_trained_HMM$delta,
        gamma = m_trained_HMM$gamma,
        distribution_class = m_trained_HMM$distribution_class,
        distribution_theta = m_trained_HMM$distribution_theta,
        decoding_method = "global")

# Globally most likely sequence of hidden states,
# i.e. in this case sequence of activity levels

global_decoding$decoding
plot(global_decoding$decoding)

# Plot the observed impulse counts and the most likely
# sequence (green) according to the Viterbi algorithm that
# generated these observations

plot(x)
lines(global_decoding$decoding_distr_means, col = "green")

# iii) Local decoding
# Decode the trained HMM using the local decoding algorithm
# to get the estimated sequence of hidden physical activity levels

local_decoding <-
   HMM_decoding(
      x = x,
      m = m_trained_HMM$m,
      delta = m_trained_HMM$delta,
      gamma = m_trained_HMM$gamma,
      distribution_class = m_trained_HMM$distribution_class,
      distribution_theta = m_trained_HMM$distribution_theta,
      decoding_method = "local")

 # Locally most likely sequence of hidden states,
 # i.e. in this case sequence of activity levels
 # local_decoding$decoding
 plot(local_decoding$decoding)

 # Plot the observed impulse counts and the most likely
 # sequence (green) according to the local decoding algorithm
 # that generated these observations
 plot(x)
 lines(local_decoding$decoding_distr_means, col = "red")

 # iv) Comparison of global and local decoding -----
```

```
# Comparison of global decoding (green), local decoding (red)
# and the connection to the closest mean (blue)
print(global_decoding$decoding)
print(local_decoding$decoding)

# Plot comparison
par(mfrow = c(2,2))
plot(global_decoding$decoding[seq(230,260)], col = "green",
     ylab = "global decoding",
     main = "(zooming)")
plot(x[seq(230,260)], ylab = "global decoding",
main = "(zooming x[seq(230,260)])")
lines(global_decoding$decoding_distr_means[seq(230,260)], col = "green")
plot(local_decoding$decoding[seq(230,260)], col = "red",
     ylab = "local decoding", main = "(zooming)")
plot(x[seq(230,260)], ylab = "local decoding",
main = "(zooming x[seq(230,260)])")
lines(local_decoding$decoding_distr_means[seq(230,260)], col = "red")

par(old.par)
```

`HMM_simulation` *Generating Realizations of a Hidden Markov Model*

Description

This function generates a sequence of hidden states of a Markov chain and a corresponding parallel sequence of observations.

Usage

```
HMM_simulation(
  size,
  m,
  delta = rep(1/m, times = m),
  gamma = 0.8 * diag(m) + rep(0.2/m, times = m),
  distribution_class,
  distribution_theta,
  obs_range = c(NA, NA),
  obs_round = FALSE,
  obs_non_neg = FALSE,
  plotting = 0
)
```

Arguments

`size`	length of the time-series of hidden states and observations (also `T`).
`m`	a (finite) number of states in the hidden Markov chain.

`delta` a vector object containing values for the marginal probability distribution of the `m` states of the Markov chain at the time point `t=1`. Default is `delta = rep(1 / m, times = m)`.

`gamma` a matrix (`ncol = nrow = m`) containing values for the transition matrix of the hidden Markov chain. Default is `gamma=0.8 * diag(m) + rep(0.2 / m, times = m)`

`distribution_class`
a single character string object with the abbreviated name of the `m` observation distributions of the Markov dependent observation process. The following distributions are supported by this algorithm: Poisson (`pois`); generalized Poisson (`genpois`); normal (`norm`); geometric (`geom`).

`distribution_theta`
a list object containing the parameter values for the `m` observation distributions that are dependent on the hidden Markov state.

`obs_range` a vector object specifying the range for the observations to be generated. For instance, the vector `c(0,1500)` allows only observations between 0 and 1500 to be generated by the HMM. Default value is `FALSE`. See Notes for further details.

`obs_round` a logical object. `TRUE` if all generated observations are natural. Default value is `FALSE`. See Notes for further details.

`obs_non_neg` a logical object. `TRUE`, if non negative observations are generated. Default value is `FALSE`. See Notes for further details.

`plotting` a numeric value between 0 and 5 (generates different outputs). NA suppresses graphical output. Default value is `0`.
`0`: output 1-5
`1`: summary of all results
`2`: generated time series of states of the hidden Markov chain
`3`: means (of the observation distributions, which depend on the states of the Markov chain) along the time series of states of the hidden Markov chain
`4`: observations along the time series of states of the hidden Markov chain
`5`: simulated observations

Value

The function `HMM_simulation` returns a list containing the following components:

The function `HMM_simulation` returns a list containing the following components:

size length of the generated time-series of hidden states and observations.

m input number of states in the hidden Markov chain.

delta a vector object containing the chosen values for the marginal probability distribution of the `m` states of the Markov chain at the time point `t=1`.

gamma a matrix containing the chosen values for the transition matrix of the hidden Markov chain.

distribution_class a single character string object with the abbreviated name of the chosen observation distributions of the Markov dependent observation process.

distribution_theta a list object containing the chosen values for the parameters of the `m` observation distributions that are dependent on the hidden Markov state.

markov_chain a vector object containing the generated sequence of states of the hidden Markov chain of the HMM.

means_along_markov_chain a vector object containing the sequence of means (of the state dependent distributions) corresponding to the generated sequence of states.

observations a vector object containing the generated sequence of (state dependent) observations of the HMM.

Note

Some notes regarding the default values:
`gamma`:
The default setting assigns higher probabilities for remaining in a state than c hanging into another.
`obs_range`:
Has to be used with caution. since it manipulates the results of the HMM. If a value for an observation at time `t` is generated outside the defined range, it will be regenerated as long as it falls into `obs_range`. It is possible just to define one boundary, e.g. `obs_range=c(NA,2000)` for observations lower than 2000, or `obs_range=c(100,NA)` for observations higher than 100. `obs_round` :
Has to be used with caution! Rounds each generated observation and hence manipulates the results of the HMM (important for the normal distribution based HMM). `obs_ non_neg`:
Has to be used with caution, since it manipulates the results of the HMM. If a negative value for an observation at a time `t` is generated, it will be re-generated as long as it is non-negative (important for the normal distribution based HMM).

Author(s)

Vitali Witowski (2013).

See Also

AIC_HMM, BIC_HMM, HMM_training

Examples

```
# i.) Generating a HMM with Poisson-distributed data -----

 Pois_HMM_data <-
   HMM_simulation(size = 300,
                  m = 5,
    distribution_class = "pois",
    distribution_theta = list( lambda=c(10,15,25,35,55)))

 print(Pois_HMM_data)


 # ii.) Generating 6 physical activities with normally -----
 #      distributed accelerometer counts using a HMM.

 # Define number of time points (1440 counts equal 6 hours of
 # activity counts assuming an epoch length of 15 seconds).
 size <- 1440
```

```
# Define 6 possible physical activity ranges
m <- 6

# Start with the lowest possible state
# (in this case with the lowest physical activity)
delta <- c(1, rep(0, times = (m - 1)))

# Define transition matrix to generate according to a
# specific activity
gamma <- 0.935 * diag(m) + rep(0.065 / m, times = m)

# Define parameters
# (here: means and standard deviations for m=6 normal
#  distributions that define the distribution in
#  a phsycial acitivity level)
distribution_theta <- list(mean = c(0,100,400,600,900,1200),
                           sd = rep(x = 200, times = 6))

# Assume for each count an upper boundary of 2000
obs_range <-c(NA,2000)

# Accelerometer counts shall not be negative
obs_non_neg <-TRUE

# Start simulation

accelerometer_data <-
    HMM_simulation(size = size,
                       m = m,
                   delta = delta,
                   gamma = gamma,
     distribution_class = "norm",
     distribution_theta = distribution_theta,
              obs_range = obs_range,
            obs_non_neg = obs_non_neg,
               plotting = 0)
print(accelerometer_data)
```

`HMM_training` *Training of Hidden Markov Models*

Description

Function to estimate the model specific parameters (`delta`, `gamma`, `distribution_theta`) for a hidden Markov model, given a time-series and a user-defined distribution class. Can also be used for model selection (selecting the optimal number of states `m`). See Details for more information.

Usage

```
HMM_training(
  x,
  distribution_class,
  min_m = 2,
  max_m = 6,
  n = 100,
  training_method = "EM",
  discr_logL = FALSE,
  discr_logL_eps = 0.5,
  Mstep_numerical = FALSE,
  dynamical_selection = TRUE,
  BW_max_iter = 50,
  BW_limit_accuracy = 0.001,
  BW_print = TRUE,
  DNM_max_iter = 50,
  DNM_limit_accuracy = 0.001,
  DNM_print = 2
)
```

Arguments

x	a vector object of length T containing observations of a time-series x, which are assumed to be realizations of the (hidden Markov state dependent) observation process of the HMM.
distribution_class	a single character string object with the abbreviated name of the m observation distributions of the Markov dependent observation process. The following distributions are supported: Poisson (pois); generalized Poisson (genpois, only available for training_method="numerical"); normal (norm)).
min_m	minimum number of hidden states in the hidden Markov chain. Default value is 2.
max_m	maximum number of hidden states in the hidden Markov chain. Default value is 6.
n	a single numerical value specifying the number of samples to find the best starting values for the training algorithm. Default value is n=100.
training_method	a logical value indicating whether the Baum-Welch algorithm ("EM") or the method of direct numerical maximization ("numerical") should be applied for estimating the model specific parameters. See Baum_Welch_algorithm and direct_numerical_maximization for further details.
discr_logL	a logical object. Default is FALSE for the general log-likelihood, TRUE for the discrete log-likelihood (for distribution_class = "norm").
discr_logL_eps	a single numerical value, used to approximate the discrete log-likelihood for a hidden Markov model based on nomal distributions (for "norm"). The default value is 0.5.

`Mstep_numerical`
: a logical object indicating whether the Maximization Step of the Baum-Welch algorithm should be performed by numerical maximization. Default is `FALSE`.

`dynamical_selection`
: a logical value indicating whether the method of dynamical initial parameter selection should be applied (see Details). Default is `TRUE`.

`BW_max_iter`
: a single numerical value representing the maximum number of iterations in the Baum-Welch algorithm. Default value is `50`.

`BW_limit_accuracy`
: a single numerical value representing the convergence criterion of the Baum-Welch algorithm. Default value is is `0.001`.

`BW_print`
: a logical object indicating whether the log-likelihood at each iteration-step shall be printed. Default is `TRUE`.

`DNM_max_iter`
: a single numerical value representing the maximum number of iterations of the numerical maximization using the nlm-function (used to perform the Maximization Step of the Baum-Welch-algorithm). Default value is `50`.

`DNM_limit_accuracy`
: a single numerical value representing the convergence criterion of the numerical maximization algorithm using the `nlm` function (used to perform the Maximization Step of the Baum-Welch- algorithm). Default value is `0.001`.

`DNM_print`
: a single numerical value to determine the level of printing of the `nlm`-function. See `nlm`-function for further informations. The value `0` suppresses, that no printing will be outputted. Default value is `2` for full printing.

Details

More precisely, the function works as follows:
Step 1: In a first step, the algorithm estimates the model specific parameters for different values of `m` (indeed for `min_m, ..., max_m`) using either the function `Baum_Welch_algorithm` or `direct_numerical_maximization`. Therefore, the function first searches for plausible starting values by using the function `initial_parameter_training`. **Step 2:** In a second step, this function evaluates the AIC and BIC values for each HMM (built in Step 1) using the functions `AIC_HMM` and `BIC_HMM`. Then, based on that values, this function decides for the most plausible number of states `m` (respectively for the most appropriate HMM for the given time-series of observations). In case when AIC and BIC claim for a different `m`, the algorithm decides for the smaller value for `m` (with the background to have a more simplistic model). If the user is intereseted in having a HMM with a fixed number for `m`, `min_m` and `max_m` have to be chosen equally (for instance `min_m=4` = `max_m` for a HMM with `m=4` hidden states). To speed up the parameter estimation for each $m > m_min$, the user can choose the method of dynamical initial parameter selection. If the method of dynamical intial parameter selection **is not applied**, the function `initial_parameter_training` will be called to find plausible starting values for each state $m \in \{min_m, \ldots, max_m\}$.
If the method of dynamical intial parameter selection **is applied**, then starting parameter values using the function `initial_parameter_training` will be found only for the first HMM (respectively the HMM with `m_min` states). The further starting parameter values for the next HMM (with `m+1` states and so on) are retained from the trained parameter values of the last HMM (with `m` states and so on).

Value

HMM_training returns a list containing the following components:

trained_HMM_with_selected_m a list object containing the key data of the optimal trained HMM (HMM with selected m) – summarized output of the Baum_Welch_algorithm or direct_numerical_maximization algorithm, respectively.

list_of_all_initial_parameters a list object containing the plausible starting values for all HMMs (one for each state m).

list_of_all_trained_HMMs a list object containing all trained m-state-HMMs. See Baum_Welch_algorithm or direct_numerical_maximization for training_method="EM" or training_method="numerical", respectively.

list_of_all_logLs_for_each_HMM_with_m_states a list object containing all logarithmized Likelihoods of each trained HMM.

list_of_all_AICs_for_each_HMM_with_m_states a list object containing the AIC values of all trained HMMs.

list_of_all_BICs_for_each_HMM_with_m_states a list object containing the BIC values of all trained HMMs.

model_selection_over_AIC is logical. TRUE, if model selection was based on AIC and FALSE, if model selection was based on BIC.

Author(s)

Vitali Witowski (2013)

References

MacDonald, I. L., Zucchini, W. (2009) *Hidden Markov Models for Time Series: An Introduction Using R*, Boca Raton: Chapman & Hall.

See Also

initial_parameter_training, Baum_Welch_algorithm, direct_numerical_maximization, AIC_HMM, BIC_HMM

Examples

```
x <- c(1,16,19,34,22,6,3,5,6,3,4,1,4,3,5,7,9,8,11,11,
  14,16,13,11,11,10,12,19,23,25,24,23,20,21,22,22,18,7,
  5,3,4,3,2,3,4,5,4,2,1,3,4,5,4,5,3,5,6,4,3,6,4,8,9,12,
  9,14,17,15,25,23,25,35,29,36,34,36,29,41,42,39,40,43,
  37,36,20,20,21,22,23,26,27,28,25,28,24,21,25,21,20,21,
  11,18,19,20,21,13,19,18,20,7,18,8,15,17,16,13,10,4,9,
  7,8,10,9,11,9,11,10,12,12,5,13,4,6,6,13,8,9,10,13,13,
  11,10,5,3,3,4,9,6,8,3,5,3,2,2,1,3,5,11,2,3,5,6,9,8,5,
  2,5,3,4,6,4,8,15,12,16,20,18,23,18,19,24,23,24,21,26,
  36,38,37,39,45,42,41,37,38,38,35,37,35,31,32,30,20,39,
  40,33,32,35,34,36,34,32,33,27,28,25,22,17,18,16,10,9,
  5,12,7,8,8,9,19,21,24,20,23,19,17,18,17,22,11,12,3,9,
```

```
  10,4,5,13,3,5,6,3,5,4,2,5,1,2,4,4,3,2,1)

# Train a poisson hidden Markov model using the Baum-Welch
# algorithm for different number of states m=2,...,6

 trained_HMMs <-
  HMM_training(x = x,
           min_m = 2,
           max_m = 6,
      distribution_class = "pois",
      training_method = "EM")

# Various output values for the HMM
names(trained_HMMs)

# Print details of the most plausible HMM for the given
# time-series of observations
print(trained_HMMs$trained_HMM_with_selected_m)

# Print details of all trained HMMs (by this function)
# for the given time-series of observations
print(trained_HMMs$list_of_all_trained_HMMs)

# Print the BIC-values of all trained HMMs for the given
# time-series of observations
print(trained_HMMs$list_of_all_BICs_for_each_HMM_with_m_states)

# Print the logL-values of all trained HMMs for the
# given time-series of observations
print(trained_HMMs$list_of_all_logLs_for_each_HMM_with_m_states)
```

`initial_parameter_training` *Algorithm to Find Plausible Starting Values for Parameter Estimation*

Description

The function computes plausible starting values for both the Baum-Welch algorithm and the algorithm for directly maximizing the log-Likelihood. Plausible starting values can potentially diminish problems of (i) numerical instability and (ii) not finding the global optimum.

Usage

```
initial_parameter_training(
  x,
  m,
  distribution_class,
```

```
  n = 100,
  discr_logL = FALSE,
  discr_logL_eps = 0.5
)
```

Arguments

x	a vector object containing the time-series of observations that are assumed to be realizations of the (hidden Markov state dependent) observation process of the model.
m	a (finite) number of states in the hidden Markov chain.
distribution_class	a single character string object with the abbreviated name of the m observation distributions of the Markov dependent observation process. The following distributions are supported: Poisson (pois); generalized Poisson (genpois, only available for training_method="numerical"); normal (norm)).
n	a single numerical value specifying the number of samples to find the best starting value for the training algorithm. Default value is 100.
discr_logL	a logical object. Default is FALSE for the general log-likelihood, TRUE for the discrete log-likelihood (for distribution_class = "norm").
discr_logL_eps	a single numerical value, used to approximate the discrete log-likelihood for a hidden Markov model based on nomal distributions (for "norm"). The default value is 0.5.

Details

From our experience, parameter estimation for long time-series of observations (T>1000) or observation values >1500 tend to be numerical unstable and does not necessarily find a global maximum. Both problems can eventually be diminished with plausible starting values. Basically, the idea behind initial_parameter_training is to sample randomly n sets of m observations from the time-series x, as means (E) of the state-dependent distributions. This n samplings of E, therefore induce n sets of parameters (distribution_theta) for the HMM without running a (slow) parameter estimation algorithm. Furthermore, initial_parameter_training calculates the log-Likelihood for all those n sets of parameters. The set of parameters with the best Likelihood are outputted as plausible starting values.

(Additionally to the n sets of randomly chosen observations as means, the m quantiles of the observations are also checked as plausible means within this algorithm.)

Value

The function initial_parameter_training returns a list containing the following components:

m input number of states in the hidden Markov chain.

k a single numerical value representing the number of parameters of the defined distribution class of the observation process.

logL logarithmized likelihood of the model evaluated at the HMM with given starting values (delta, gamma, distribution theta) induced by E.

E randomly choosen means of the observation time-series `x`, used for the observation distributions, for which the induced parameters (`delta, gamma, distribution theta`) produce the largest Likelihood.

distribution_theta a list object containing the plausible starting values for the parameters of the `m` observation distributions that are dependent on the hidden Markov state.

delta a vector object containing plausible starting values for the marginal probability distribution of the `m` states of the Markov chain at the time point `t=1`. At the moment:
`delta = rep(1/m, times=m)`.

gamma a matrix (`nrow=ncol=m`) containing the plausible starting values for the transition matrix of the hidden Markov chain. At the moment:
`gamma = 0.8 * diag(m) + rep(0.2/m, times=m)`.

Author(s)

Vitali Witowski (2013).

See Also

Baum_Welch_algorithm, direct_numerical_maximization, HMM_training

Examples

```
x <- c(1,16,19,34,22,6,3,5,6,3,4,1,4,3,5,7,9,8,11,11,
  14,16,13,11,11,10,12,19,23,25,24,23,20,21,22,22,18,7,
  5,3,4,3,2,3,4,5,4,2,1,3,4,5,4,5,3,5,6,4,3,6,4,8,9,12,
  9,14,17,15,25,23,25,35,29,36,34,36,29,41,42,39,40,43,
  37,36,20,20,21,22,23,26,27,28,25,28,24,21,25,21,20,21,
  11,18,19,20,21,13,19,18,20,7,18,8,15,17,16,13,10,4,9,
  7,8,10,9,11,9,11,10,12,12,5,13,4,6,6,13,8,9,10,13,13,
  11,10,5,3,3,4,9,6,8,3,5,3,2,2,1,3,5,11,2,3,5,6,9,8,5,
  2,5,3,4,6,4,8,15,12,16,20,18,23,18,19,24,23,24,21,26,
  36,38,37,39,45,42,41,37,38,38,35,37,35,31,32,30,20,39,
  40,33,32,35,34,36,34,32,33,27,28,25,22,17,18,16,10,9,
  5,12,7,8,8,9,19,21,24,20,23,19,17,18,17,22,11,12,3,9,
  10,4,5,13,3,5,6,3,5,4,2,5,1,2,4,4,3,2,1)

# Finding plausibel starting values for the parameter estimation
# for a generealized-Pois-HMM with m=4 states

m <- 4

 plausible_starting_values <-
 initial_parameter_training(x = x,
                            m = m,
           distribution_class = "genpois",
                            n = 100)

 print(plausible_starting_values)
```

`local_decoding_algorithm` *Algorithm for Decoding Hidden Markov Models (local)*

Description

The function decodes a hidden Markov model into a most likely sequence of hidden states. Different to the `Viterbi_algorithm`, this algorithm determines the most likely hidden state for each time point seperately.

Usage

```
local_decoding_algorithm(
  x,
  m,
  delta,
  gamma,
  distribution_class,
  distribution_theta,
  discr_logL = FALSE,
  discr_logL_eps = 0.5
)
```

Arguments

`x`	a vector object containing the time-series of observations that are assumed to be realizations of the (hidden Markov state dependent) observation process of the model.
`m`	a (finite) number of states in the hidden Markov chain.
`delta`	a vector object containing values for the marginal probability distribution of the `m` states of the Markov chain at the time point `t=1`.
`gamma`	a matrix (`ncol=nrow=m`) containing values for the transition matrix of the hidden Markov chain.
`distribution_class`	a single character string object with the abbreviated name of the `m` observation distributions of the Markov dependent observation process. The following distributions are supported by this algorithm: Poisson (`pois`); generalized Poisson (`genpois`); normal (`norm`); geometric (`geom`).
`distribution_theta`	a list object containing the parameter values for the `m` observation distributions that are dependent on the hidden Markov state.
`discr_logL`	a logical object. It is `TRUE` if the discrete log-likelihood shall be calculated (for `distribution_class="norm"` instead of the general log-likelihood). Default is `FALSE`.
`discr_logL_eps`	a single numerical value to approximately determine the discrete log-likelihood for a hidden Markov model based on nomal distributions (for `"norm"`). The default value is `0.5`.

Value

`x` a vector object containing the time-series of observations that are assumed to be realizations of the (hidden Markov state dependent) observation process of the model.

`m` a (finite) number of states in the hidden Markov chain.

`delta` a vector object containing values for the marginal probability distribution of the `m` states of the Markov chain at the time point `t=1`.

`gamma` a matrix (`ncol=nrow=m`) containing values for the transition matrix of the hidden Markov chain.

`distribution_class`
a single character string object with the abbreviated name of the `m` observation distributions of the Markov dependent observation process. The following distributions are supported by this algorithm: Poisson (`pois`); generalized Poisson (`genpois`); normal (`norm`); geometric (`geom`).

`distribution_theta`
a list object containing the parameter values for the `m` observation distributions that are dependent on the hidden Markov state.

`discr_logL` a logical object. It is `TRUE` if the discrete log-likelihood shall be calculated (for `distribution_class="norm"` instead of the general log-likelihood). Default is `FALSE`.

`discr_logL_eps` a single numerical value to approximately determine the discrete log-likelihood for a hidden Markov model based on nomal distributions (for `"norm"`). The default value is `0.5`.

Author(s)

The basic algorithm for a Poisson-HMM can be found in MacDonald & Zucchini (2009, Paragraph A.2.6). Extension and implementation by Vitali Witowski (2013).

References

MacDonald, I. L., Zucchini, W. (2009) *Hidden Markov Models for Time Series: An Introduction Using R*, Boca Raton: Chapman & Hall.

See Also

`Viterbi_algorithm`, `HMM_decoding`

Examples

```
x <- c(1,16,19,34,22,6,3,5,6,3,4,1,4,3,5,7,9,8,11,11,
  14,16,13,11,11,10,12,19,23,25,24,23,20,21,22,22,18,7,
  5,3,4,3,2,3,4,5,4,2,1,3,4,5,4,5,3,5,6,4,3,6,4,8,9,12,
  9,14,17,15,25,23,25,35,29,36,34,36,29,41,42,39,40,43,
  37,36,20,20,21,22,23,26,27,28,25,28,24,21,25,21,20,21,
  11,18,19,20,21,13,19,18,20,7,18,8,15,17,16,13,10,4,9,
  7,8,10,9,11,9,11,10,12,12,5,13,4,6,6,13,8,9,10,13,13,
  11,10,5,3,3,4,9,6,8,3,5,3,2,2,1,3,5,11,2,3,5,6,9,8,5,
```

```
  2,5,3,4,6,4,8,15,12,16,20,18,23,18,19,24,23,24,21,26,
  36,38,37,39,45,42,41,37,38,38,35,37,35,31,32,30,20,39,
  40,33,32,35,34,36,34,32,33,27,28,25,22,17,18,16,10,9,
  5,12,7,8,8,9,19,21,24,20,23,19,17,18,17,22,11,12,3,9,
  10,4,5,13,3,5,6,3,5,4,2,5,1,2,4,4,3,2,1)

# Train hidden Markov model for m = 4

 m_trained_HMM <-
   HMM_training(x = x,
            min_m = 4,
            max_m = 4,
     distribution_class = "pois")$trained_HMM_with_selected_m

# Decode the trained HMM using the local-decoding algorithm
# to get the locally most likely sequence of hidden states
# for the time-series of observations
local_decoding <-
  local_decoding_algorithm(
        x = x,
        m = m_trained_HMM$m,
        delta = m_trained_HMM$delta,
        gamma = m_trained_HMM$gamma,
        distribution_class = m_trained_HMM$distribution_class,
        distribution_theta = m_trained_HMM$distribution_theta)

# Most likely sequence of hidden states
print(local_decoding$decoding)
plot(local_decoding$decoding)
```

`pgenpois` *The Generalized Poisson Distribution*

Description

Distribution function for the generalized Poisson distribution.

Usage

```
pgenpois(q, lambda1, lambda2)
```

Arguments

`q`	a numeric vector of quantiles
`lambda1`	a single numeric value for parameter `lambda1` with $lambda1 > 0$
`lambda2`	a single numeric value for parameter `lambda2` with $0 \leq lamdba2 < 1$. When `lambda2=0`, the generalized Poisson distribution reduces to the Poisson distribution

Details

The generalized Poisson distribution has the density

$$p(x) = \lambda_1(\lambda_1 + \lambda_2 \cdot x)^{x-1} \frac{\exp(-\lambda_1 - \lambda_2 \cdot x))}{x!}$$

for $x = 0, 1, 2, \ldots$,b with $\mathrm{E}(X) = \frac{\lambda_1}{1-\lambda_2}$ and variance $\mathrm{var}(X) = \frac{\lambda_1}{(1-\lambda_2)^3}$.

Value

pgenpois gives the distribution function of the generalized Poisson distribution.

Author(s)

Based on Joe and Zhu (2005). Implementation by Vitali Witowski (2013).

References

Joe, H., Zhu, R. (2005). Generalized poisson distribution: the property of mixture of poisson and comparison with negative binomial distribution. Biometrical Journal **47**(2):219–229.

See Also

pgenpois, rgenpois; Distributions for other standard distributions, including dpois for the Poisson distribution.

Examples

```
dgenpois(x = seq(0,20), lambda1 = 10, lambda2 = 0.5)
pgenpois(q = 5, lambda1 = 10, lambda2 = 0.5)
hist(rgenpois(n = 1000, lambda1 = 10, lambda2 = 0.5) )
```

---

rgenpois *The Generalized Poisson Distribution*

Description

Density, distribution function and random generation function for the generalized Poisson distribution.

Usage

```
rgenpois(n, lambda1, lambda2)
```

Arguments

n	number of observations
lambda1	a single numeric value for parameter lambda1 with $lambda1 > 0$
lambda2	a single numeric value for parameter lambda2 with $0 \leq lamdba2 < 1$. When lambda2=0, the generalized Poisson distribution reduces to the Poisson distribution

Details

The generalized Poisson distribution has the density

$$p(x) = \lambda_1(\lambda_1 + \lambda_2 \cdot x)^{x-1} \frac{\exp(-\lambda_1 - \lambda_2 \cdot x))}{x!}$$

for $x = 0, 1, 2, \ldots$,b with $\mathrm{E}(X) = \frac{\lambda_1}{1-\lambda_2}$ and variance $\mathrm{var}(X) = \frac{\lambda_1}{(1-\lambda_2)^3}$.

Value

rgenpois generates random deviates of the generalized Poisson distribution.

Author(s)

Based on Joe and Zhu (2005). Implementation by Vitali Witowski (2013).

References

Joe, H., Zhu, R. (2005). Generalized poisson distribution: the property of mixture of poisson and comparison with negative binomial distribution. Biometrical Journal **47**(2):219–229.

See Also

pgenpois, dgenpois; Distributions for other standard distributions, including dpois for the Poisson distribution.

Examples

```
dgenpois(x = seq(0,20), lambda1 = 10, lambda2 = 0.5)
pgenpois(q = 5, lambda1 = 10, lambda2 = 0.5)
hist(rgenpois(n = 1000, lambda1 = 10, lambda2 = 0.5) )
```

`Viterbi_algorithm` *Algorithm for Decoding Hidden Markov Models (global)*

Description

The function decodes a trainded hidden Markov model into a most likely sequence of hidden states. Different to the `local_decoding_algorithm`, this algorithm determines the sequence of most likely hidden states for all time points simultaneously. See MacDonald & Zucchini (2009, Paragraph 5.3.2) for further details.

Usage

```
Viterbi_algorithm(
  x,
  m,
  delta,
  gamma,
  distribution_class,
  distribution_theta,
  discr_logL = FALSE,
  discr_logL_eps = 0.5
)
```

Arguments

`x`	a vector object containing the time-series of observations that are assumed to be realizations of the (hidden Markov state dependent) observation process of the model.
`m`	a (finite) number of states in the hidden Markov chain.
`delta`	a vector object containing values for the marginal probability distribution of the `m` states of the Markov chain at the time point `t=1`.
`gamma`	a matrix (`ncol=nrow=m`) containing values for the transition matrix of the hidden Markov chain.
`distribution_class`	a single character string object with the abbreviated name of the `m` observation distributions of the Markov dependent observation process. The following distributions are supported by this algorithm: Poisson (`pois`); generalized Poisson (`genpois`); normal (`norm`); geometric (`geom`).
`distribution_theta`	a list object containing the parameter values for the `m` observation distributions that are dependent on the hidden Markov state.
`discr_logL`	a logical object. It is `TRUE` if the discrete log-likelihood shall be calculated (for `distribution_class="norm"` instead of the general log-likelihood). Default is `FALSE`.
`discr_logL_eps`	a single numerical value to approximately determine the discrete log-likelihood for a hidden Markov model based on nomal distributions (for `"norm"`). The default value is `0.5`.

Value

The `Viterbi_algorithm` returns a list containing the following two components:

omega a (T,m)-matrix (when T indicates the length/size of the observation time-series and m the number of states of the HMM) containing probabilities (maximum probability to generate the first t members (t=1,...,T) of the given time-series x with the HMM and to stop in state i=1,...,m) calculated by the algorithm. See MacDonald & Zucchini (2009, Paragraph 5.3.2) for further details.

decoding a numerical vector containing the globally most likely sequence of hidden states as decoded by the Viterbi algorithm.

Author(s)

The basic algorithm for a Poisson-HMM can be found in MacDonald & Zucchini (2009, Paragraph A.2.4). Extension and implementation by Vitali Witowski (2013).

References

MacDonald, I. L., Zucchini, W. (2009) *Hidden Markov Models for Time Series: An Introduction Using R*, Boca Raton: Chapman & Hall.

Forney, G.D. (1973). The Viterbi algorithm. Proceeding of the IEE, vol. **61**(3), 268–278.

Viterbi, A.J. (1967). Error Bounds for concolutional codes and an asymptotically optimal decoding algorithm. Information Theory, IEEE Transactions on, vol. **13**(2), 260–269.

See Also

`local_decoding_algorithm`, `HMM_decoding`

Examples

```
x <- c(1,16,19,34,22,6,3,5,6,3,4,1,4,3,5,7,9,8,11,11,
  14,16,13,11,11,10,12,19,23,25,24,23,20,21,22,22,18,7,
  5,3,4,3,2,3,4,5,4,2,1,3,4,5,4,5,3,5,6,4,3,6,4,8,9,12,
  9,14,17,15,25,23,25,35,29,36,34,36,29,41,42,39,40,43,
  37,36,20,20,21,22,23,26,27,28,25,28,24,21,25,21,20,21,
  11,18,19,20,21,13,19,18,20,7,18,8,15,17,16,13,10,4,9,
  7,8,10,9,11,9,11,10,12,12,5,13,4,6,6,13,8,9,10,13,13,
  11,10,5,3,3,4,9,6,8,3,5,3,2,2,1,3,5,11,2,3,5,6,9,8,5,
  2,5,3,4,6,4,8,15,12,16,20,18,23,18,19,24,23,24,21,26,
  36,38,37,39,45,42,41,37,38,38,35,37,35,31,32,30,20,39,
  40,33,32,35,34,36,34,32,33,27,28,25,22,17,18,16,10,9,
  5,12,7,8,8,9,19,21,24,20,23,19,17,18,17,22,11,12,3,9,
  10,4,5,13,3,5,6,3,5,4,2,5,1,2,4,4,3,2,1)

# Train hidden Markov model for m = 4 -----

  m_trained_HMM <-
     HMM_training(x = x,
                  min_m = 4,
                  max_m = 4,
```

```
 distribution_class = "pois")$trained_HMM_with_selected_m

 # Decode the trained HMM using the Viterbi algorithm to get
 # the globally most likely sequence of hidden states for
 # the time-series of observations
 global_decoding <-
 Viterbi_algorithm(x = x,
                   m = m_trained_HMM$m,
               delta = m_trained_HMM$delta,
               gamma = m_trained_HMM$gamma,
     distribution_class = m_trained_HMM$distribution_class,
     distribution_theta = m_trained_HMM$distribution_theta)

# Most likely sequence of hidden states

print(global_decoding$decoding)
plot(global_decoding$decoding)
```

Index